

# Improving the Electric Circuit Simulator using Homotopy Methods

---

Jatin Vikram Singh  
Indian Institute of Technology Kanpur, India

Advisor: Professor Ljiljana Trajkovic  
School of Engineering Science  
Simon Fraser University Burnaby, BC, Canada



# Outline

---

- Why Simulate!!!
- DC Operating Points
- Homotopy Methods
- Modified Nodal Analysis
- The Parser : Platform Change and Improvements
- Conclusion

# Why Simulate!!!

---

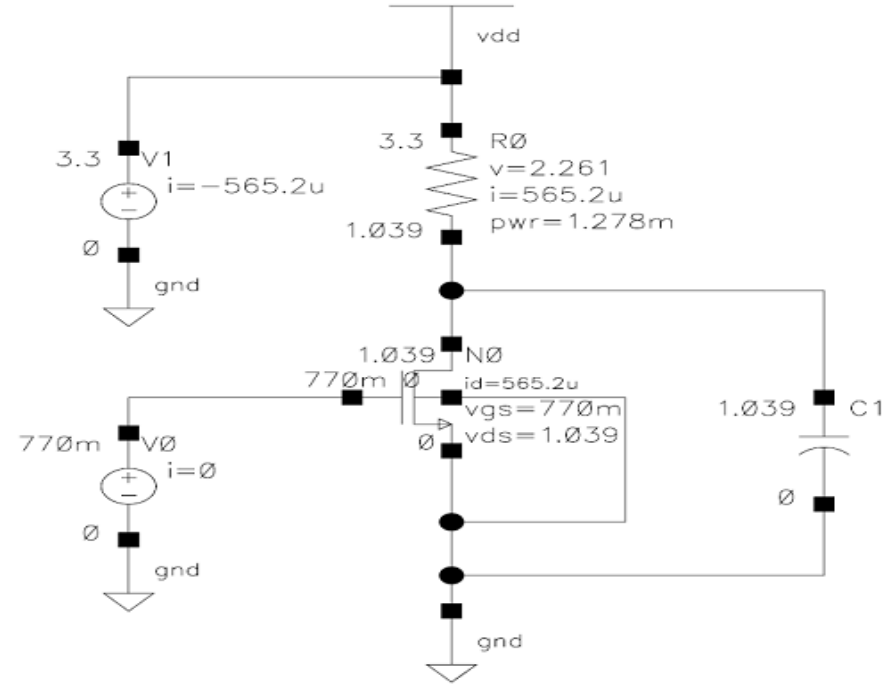
- Understand principles of system
- Propose solutions to problems
- Vary the model to meet demands as required
- Saves us a lot of resources required to build these devices and money as well

The logo for PSpice, featuring the word "PSPICE" in a bold, blue, sans-serif font with a red outline and a slight 3D effect.The logo for MATLAB SIMULINK, featuring a colorful, abstract graphic on the left and the text "MATLAB SIMULINK" in a black, serif font on the right.



# DC Operating Points

- Also known as Bias Points or quiescent points, are the values of voltages and currents in the DC state for the devices.
- Transistors behave differently under AC and DC sources.



# Homotopy Methods

---

- A numerical method used to find zeros of a system of equations.
- Create a simpler problem and then deform this problem into the original one.
- A series of zeros is computed from the simple problem until we find the solutions for the problem of interest.



# Homotopy Methods

---

- Given a system of equations :  $F(x) = 0$
- Generate a new function called Homotopy Function :  $H(x, \lambda) = 0$
- Varying  $\lambda$  from 0 to 1 varies homotopy function from a Gleek function  $G(x)$  to the original function  $F(x)$  such that  $H(x, 0) = G(x)$  and  $H(x, 1) = F(x)$ .
- E.g. :  $H(x, \lambda) = G(x)(1 - \lambda) + \lambda F(x)$  where  $G(x)$  could be  $(x - a)$

# Homotopy Methods

---

- The objective is to find the set :  $H^{-1}(0) = \{(x, \lambda) | H(x, \lambda) = 0\}$
- Inside this set we hope to find a continuous path which connects zeros of  $H(x, 0) = G(x)$  to  $H(x, 1) = F(x)$ .
- To trace the curve, we differentiate the homotopy function with respect to  $x$  and  $\lambda$  to create a set of differential equations.
- These equations are then solved numerically to get the solution, DC operating points to the circuit.



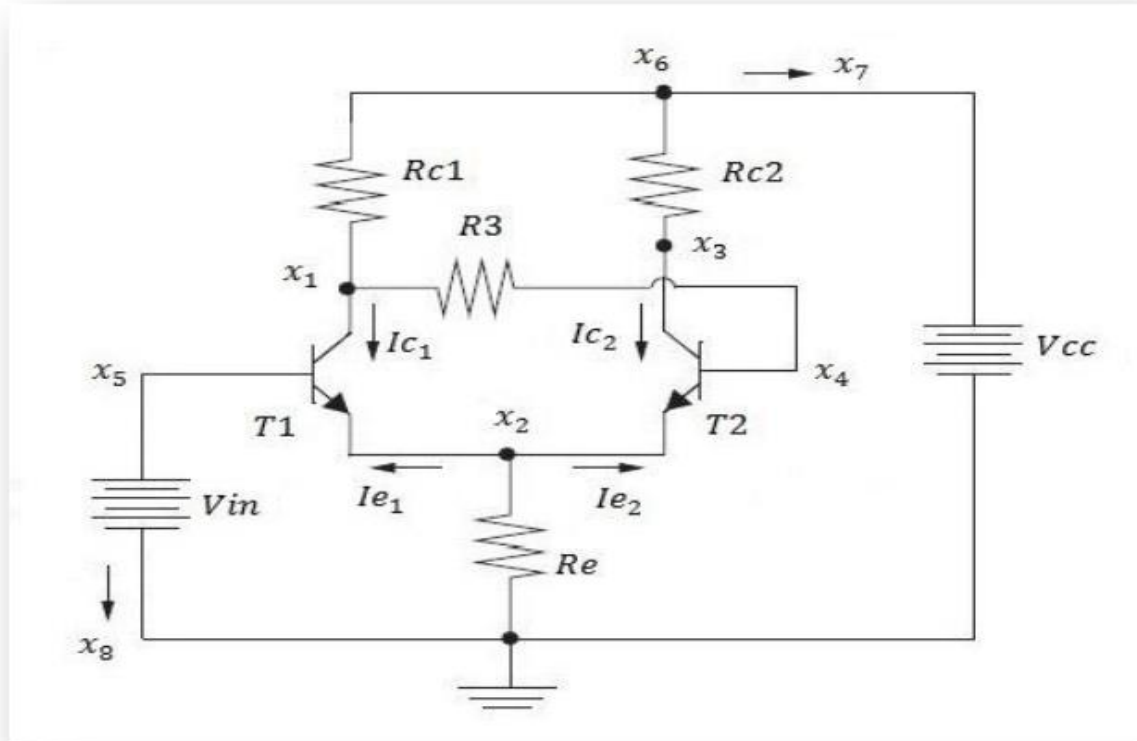
# Modified Nodal Analysis

---

- MNA often results in larger systems of equations than the other methods, but is easier to implement algorithmically on a computer which is a substantial advantage for automated solution.
- To use modified nodal analysis one equation for each node not attached to a voltage source is written (as in standard nodal analysis), and then these equations are augmented with an equation for each voltage source.
- In the figure next slide, first six equations are standard nodal analysis and the rest are additional equations to balance the number of unknowns and equations



# Modified Nodal Analysis



$$\frac{x_1 - x_4}{R3} + \frac{x_1 - x_6}{Rc1} + Ic_1 = 0$$

$$\frac{x_2}{Re} + Ie_1 + Ie_2 = 0$$

$$\frac{x_3 - x_6}{Rc2} + Ic_2 = 0$$

$$\frac{x_4 - x_1}{R3} - Ic_2 - Ie_2 = 0$$

$$x_5 - Vin = 0$$

$$x_6 - Vcc = 0$$

$$\frac{x_6 - x_1}{Rc1} + \frac{x_6 - x_3}{Rc2} + x_7 = 0$$

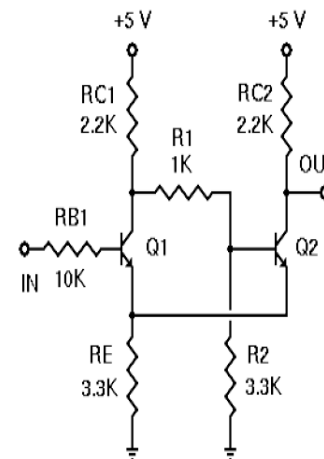
$$x_8 - Ic_1 - Ie_1 = 0$$

# The Parser

- The parser is a C++ program that takes a SPICE Netlist file as input and returns Modified Nodal Equations and Jacobians.
- This output is then used by a Matlab script to apply homotopy and find the DC operating points of the circuit.

```
Rc1 1 2 2.2K
R1 2 3 1K
Rc2 1 4 2.2K
Q1 2 5 6 Q2N2222A
Q2 4 3 6 Q2N2222A
Vin 5 0 5.0
RE 6 0 3.3K
R2 3 0 3.3K
```

```
.model Q2N2222A NPN BF=150 IS=1E-16 BR=7.5
```



Spice Netlist File



# The Parser : Platform change

```
Command Prompt
C:\MinGW\bin>g++ parser.cc -o parser
C:\MinGW\bin>parser -f Netlist_smt.txt -o Cmd_output.txt
Check POINT 1: 5
Available Equations Types Are:
<1> Nodal
<2> Modified Nodal
Please enter your choice <1, 2>:
2
Check POINT 2: Cmd_output.txt
Check POINT 3: Reached here
Output saved to file: Cmd_output.txt
Check POINT 4: .model

C:\MinGW\bin>
```



Solution1 - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP

Local Windows Debugger Debug Win32

Quick Launch (Ctrl+Q) Jatin Vikram Singh JS

Project1 parser.h parser.cpp (Global Scope) main(int argc, char \* argv[])

```
456 if (eqType != Modified){
457     compPtr = compList.getComp(0);
458     while (compPtr != NULL) {
459         eqnCount = compPtr->printSuperNode(outFile, datum, lastnode, eqnCount);
460         compPtr = compPtr->getNext();
461     }
462 }
463
464 // go down the node list and give additional MNA equations
465 if (eqType == Modified) {
466     nodePtr = nodeList.getNode(0);
467     while (nodePtr != NULL){
468         //if (nodePtr->getNameNum() != datum)
469         eqnCount = nodePtr->printMNA(outFile, datum, lastnode, eqnCount);
470         nodePtr = nodePtr->getNext();
471     }
472 }
473
474 // print jacobians
475 outFile << endl
476 << "*****" << endl;
477 outFile << endl << "          Jacobians: " << endl;
478 nodePtr1 = nodeList.getNode(0);
479
480 while (nodePtr1 != NULL) { //~> this loop handles the nodes not connected to a Vsource and those ones that are not the 'datum' node
481     //if (nodePtr1->getNameNum() != datum)
482     if (nodePtr1->getNameNum() != 0)
483     {
484         nodePtr2 = nodeList.getNode(0);
485         while (nodePtr2 != NULL) {
486             //if (nodePtr2->getNameNum() != datum)
487             if (nodePtr2->getNameNum() != 0)
488             {
489                 nodePtr1->printJac(outFile, datum, nodePtr2, lastnode, eqType);
490             }
491             nodePtr2 = nodePtr2->getNext();
492         }
493     }
494     nodePtr1 = nodePtr1->getNext();
495 }
496
497 // go down the component list and give equations for all sources
498 compPtr = compList.getComp(0);
499 while (compPtr != NULL){
500     nodePtr2 = nodeList.getNode(0);
501     compPtr2 = compList.getComp(0);
502     while (nodePtr2 != NULL){
503         //if (nodePtr2->getNameNum() != datum)
504     }
```

Output

Ready Ln 535 Col 1 Ch 1 INS

Visual Studio



# The Parser : Improvements

---

## Datum Node

- Node with maximum number of connections.
- Ground node made the datum.
- Making ground as datum fixed many equations and Jacobian errors.

## Nodal and MNA Equations

- Equations for some nodes had missing node voltages.
- Changing Datum node and fixing the equation printing module.

# The Parser : Improvements

---

## Equation Numbering

- Initially equations were being numbered by the node numbers, this caused repetition.
- They have been modified to be numbered consecutively, keeping the variables in the equations same as before.

## Jacobian

- The Jacobians adjusted after the systematic numbering of equations and changing the datum node.
- Repetitions in some Jacobian values were removed by making some modifications in the code.



# Conclusion

---

The output file is the standard format to be used by the Matlab code to employ homotopy on the circuit equations to evaluate the DC operating points. The output files have been attached along with this presentation.

Modified Nodal Analysis Equations and Jacobians		
Schmidt Trigger	<u>Eric's Output</u>	<u>Current Output</u>
Chua's Circuit	<u>Eric's Output</u>	<u>Current Output</u>

Thank You !!!!!!!

---